Definitionen für Espresso:

Ein Doppelbit ist ein Tupel aus 2 Bits und kann folgende Werte annehmen:
00 : **Inv** (= Invalid)
01 : **Zero**
10 : **One**
11 : **DC** (= Don‘t Care)

Sei *f* eine Funktion von n Variablen, dann ist ein Cube C_f von *f* ein Tupel aus n Doppelbits, von welchen keines **Inv** sein darf.

Eine Menge *S* aus Cubes heißt Set. $S_1 \circ S_2$ bedeutet hier die Konkatenation zweier Sets.

Eine Belegung ordnet jeder Eingangsvariable einen Wert aus {**Zero**, **One**} zu. Eine Belegung einer Funktion *f* von n Variablen ist folglich durch ein Tupel B_f aus n Doppelbits, die entweder **Zero** oder **One** sein dürfen, vollständig spezifiziert.

Interpretation: Ein Cube C_f beschreibt/spezifiziert eine "rechteckige" Teilmenge M aus der Menge aller Belegungen von *f* geschrieben $\mathrm{M}(C_f)$. Spezifiziere C_1 eine Menge M_1 und C_2 eine Menge M_2, dann spezifiziert das Set $S = C_1 \circ C_2$ die Menge $M_1 \cup M_2$ oder anders geschrieben:
$$\mathrm{M}(C_1 \circ C_2) = \mathrm{M}(C_1) \cup \mathrm{M}(C_2)$$

Überdeckung (Coverage): Spezifiziere S_1 die Menge M_1 und S_2 die Menge M_2, dann gilt S_1 covers S_2, wenn $M_2 \subseteq M_1$: $S_1 \,\mathrm{covers}\, S_2 \Rightarrow \mathrm{M}(S_2) \subseteq \mathrm{M}(S_1)$

Eine zweistufige, logische Funktion *f* aus n Variablen wird spezifiziert durch drei Sets ON_f, DC_f, OFF_f. Für alle Belegungen B_f: ON_f covers B_f wertet *f* zu **One** aus, für alle B_f : OFF_f covers B_f wertet *f* zu **Zero** aus. Für DC_f kann die entsprechende Bildmenge frei gewählt werden, um einen hohen Grad der Optimierung zu erreichen.
$ON_f \circ DC_f \circ OFF_f$ spezifizieren zusammen die Menge aller möglichen Belegungen von *f*.

Bitoperatoren: and, or, not sind Operatoren auf Cubes [...]

Bedeutung der and-Operation: $\mathrm{M}(C_1 \,\mathrm{and}\, C_2) = \mathrm{M}(C_1) \cap \mathrm{M}(C_2)$

Bedeutung der or-Operation: Spezifiziere C_1 die Menge M_1 und C_2 die Menge M_2, dann spezifiziert C_1 or C_2 die kleinste "rechteckige" Hülle M_3 : $M_1 \subseteq M_3 \wedge M_2 \subseteq M_3$. (Hier genauere Definition)

Enthällt ein Cube das Doppelbit **Inv**, so spezifiziert er die leere Menge: $\boldsymbol{Inv} \in C \Rightarrow \mathrm{M}(C) = \{\}$

Prüfen, ob Cube Teil eines Sets ist

Spezifiziere ein Cube *C* die Menge *M*. Eine Belegung *B* erfüllt *C* genau dann, wenn
$$B \in \mathrm{M}(C) \Leftrightarrow \mathbf{Inv} \notin B \,\mathrm{and}\, C$$

Ein Cube *U* wird von einem Set *S* überdeckt (gecovert) wenn das Hinzufügen von *U* zu *S* keine Auswirkung auf die durch *S* spezifizierte Menge hat: $S \,\mathrm{covers}\, U \Leftrightarrow \mathrm{M}(S) = \mathrm{M}(S \circ U)$

Daraus folgt, dass es eine Teilmenge S' ⊆ S geben muss, sodass:
$$(\nexists C \in S' : \mathbf{Inv} \in C) \wedge (\exists C_1 \ldots C_m \in S' : \mathrm{M}(C_1 \,\mathrm{and}\, U \circ \ldots \circ C_m \,\mathrm{and}\, U) = \mathrm{M}(U))$$
Das Ausschließen von **Inv** ist keine Notwendigkeit für die Richtigkeit der zweiten Teilaussage, reduziert im Folgenden aber den Rechenaufwand.

Es gilt:
$\mathrm{M}(C_1 \text{ or } C_2) = \mathrm{M}(C_1 \circ C_2) \Leftrightarrow \mathrm{M}(C_1 \circ C_2) = |\mathrm{M}(C_1)| + |\mathrm{M}(C_2)| - |\mathrm{M}(C_1 \text{ and } C_2)| = |\mathrm{M}(C_1 \text{ or } C_2)|$
wobei |M| die Kardinalität der Menge M ist.
Das heißt, das Zusammenfassen von zwei Cubes mit der or-Operation ist nur dann möglich, wenn das Ergebnis genau so viele Belegungen abdeckt, wie das Set aus beiden Cubes.

Die Berechnung der Kardinalität eines beliebigen Sets *S* wird schnell zu einem schwierigen Problem. Es sei denn: $\nexists C_{1,} C_2 \in S : \mathbf{Inv} \notin (C_1 \text{ and } C_2)$ ← Es gibt keine Überlappung zwischen den Cubes im Set

Annahme: Es existiert zu jedem Set *S* ein hinreichend speichereffizientes Set $\bar{S}$, welches die letztere Bedingung erfüllt (intersect-free ist). Wie ein solches Set generiert werden kann später.

Anhand dieses Sets kann effizient überprüft werden, ob ein Cube vollständig im Ursprünglichen On+Dc-Set ist:
$O\bar{n}Dc \text{ covers } U \Leftrightarrow \exists C_1 ... C_m \in O\bar{n}Dc : |\mathrm{M}(U)| = |\mathrm{M}(C_1 \text{ and } U)| + ... + |\mathrm{M}(C_m \text{ and } U)|$

Presto-Algorithmus

Dieser Algogithmus überprüft (relativ) effizent, ob ein Cube U von einem (beliebigen) Set S gecovert wird.
Dazu wird für jeden cube C ∈ S geprüft, ob dieser eine Schnittmenge mit U hat:

- Wenn diese Schnittmenge gleich U ist, so ist U vollständig von C gecovert → gebe **true** zurück
- Wenn kein C ∈ S eine Schnittmenge mit U hat, gebe **false** zurück
- Wenn mind. ein C eine Schnittmenge mit U hat, aber U von keinem C vollständig gecovert wird, teile U in zwei Hälften:
  - wähle eine Variable v, die in U = **DC** ist, aber in U and C ≠ **DC** ist.
  - Erstelle U*, U** = U
  - Setze v in U* auf **ONE**
  - Setze v in U** auf **ZERO**

  Rufe den Test rekursiv auf:
  - Wenn S U* und U** covert, gebe **true** zurück, sonst **false**.

Nach dem gleichen Prinzip kann eine add-Methode für ein Set $\bar{S}$ implementiert werden, die nur diejenigen Teile eines Cubes U zu S hinzufügt, die nicht vom aktuellen $\bar{S}$ gecovert werden:

- Füge U zu $\bar{S}$ hinzu, wenn kein C ∈ $\bar{S}$ eine Schnittmenge mit U hat
- breche ab, wenn ein C ∈ $\bar{S}$ U vollständig covert
- sonst: teile U in zwei Hälften und rufe add rekursiv auf

Complementation

Für den Expansions-Check ist es schneller zu prüfen, ob ein Cube eine Überschneidung mit dem off-Set hat, als zu testen, ob on+dc-Set den Cube vollständig covern. Dazu muss jedoch das off-Set berechnet werden (Das Complement der Funktion gebildet werden).

Hierzu kann wieder der Presto-Algorithmus verwendet werden:

- Erstelle leeres off-Set
- Erstelle einen Cube C, in dem alle Variablen **DC** sind (tautology)
- Wende das add-Prinzip zum Erstellen des intersect-freien Sets an: Füge nur diejenigen Teile von U zu S hinzu, die nicht vom on+dc-Set gecovert werden.